Semantic Interoperability

December 18, 2017

T2TRG/WISHI

Outline

- Semantic Interoperability
- Example: IPSO Temperature Object
- OCF mapping
  - Direct use of of IPSO Identifiers
  - Mapping to OCF Identifiers
  - RAML and JSON Schema
- Layered Semantic Interoperability
  - Protocol-neutral semantic vocabulary - iotschema
  - Hypermedia controls – W3C Thing Description

Semantic Interoperability

- Practical definition: Resolve "What to do" and "How to do it"
- "What to do" is a problem of horizontal normalization – of semantic concepts and meta models
- "How to do it" is a problem of vertical integration – of diverse data models and protocols
- Workflow tends to break down into discovery and configuration phase (What to do) and operation phase (How to do it)

IPSO Smart Objects

- Meta-model, based on OMA LWM2M, consisting of reusable definitions for:
  - Objects, which encapsulate some modular function like a measurement
  - Resources, which each represent a simple value that exposes or describes the function of the Object
  - An Object is a collection of Resources
- Semantic interoperability is achieved through a pre-defined set of Object and Resource identifiers
- LWM2M Protocol "baggage" and constraints

Example: IPSO Temperature Object Definition – also XML file

Object	Object ID	Object URN	Multiple Instances?	Description
IPSO Temperature	3303	urn:oma:lwm2m:ext:3303	Yes	Temperature sensor, example units = Cel

Resource Name	Resource ID	Access Type	Multiple Instances?	Mandatory	Type	Range or Enumeration	Units	Description
Sensor Value	5700	R	No	Mandatory	Float			
Units	5701	R	No	Optional	String			
Min Measured Value	5601	R	No	Optional	Float	Same as Measured Value	Same as Measured Value	
Max Measured Value	5602	R	No	Optional	Float	Same as Measured Value	Same as Measured Value	

IPSO Temperature Object Representation

```
GET /3303/0

[
 {"bn": "3303/0/" },
 {"n": "5700", "v": 27},
 {"n": "5701", "v": "Cel" },
 {"n": "5601", "v": 22 },
 {"n": "5602", "v": 40,
 {"n": "5603", "v": 0 },
 {"n": "5604", "v": 100 }
]
```

OCF

- OCF uses a meta model consisting of Resources that expose multiple Properties
- The format is a JSON-like object with named properties
- RAML and JSON Schema are used to specify the properties of a typed resource and the possible interactions as CRUD operations
- OCF uses RFC 6690 style links to index resources and create collections of resources
- OCF allows links and batch interactions on a collection resource

OCF Example – Temperature resource Re-using OCF Identifiers

GET /oic/res?if=oicif.ll

```
{
 "href": "/example/temperature",
 "if": ["oic.if.baseline","oic.if.s"],
 "rt": ["oic.r.temperature", "oic.r.minmax"]
}
```

GET /example/temperature?if=oicif.s

```
{
 "temperature": 27,
 "units": "Cel",
 "range": [0, 100],
 "minimum": 22,
 "maximum": 40,
 "resetminmax": false
}
```

Semantic Interoperability using IPSO and OCF

- Semantic model, data formats, and protocols are tightly coupled
- "What to do" and "How to do it" are defined in schemas and by type tags
- Modular and composable, but only intra-operable with compatible devices
- Device level interoperability requires choosing one format or the other

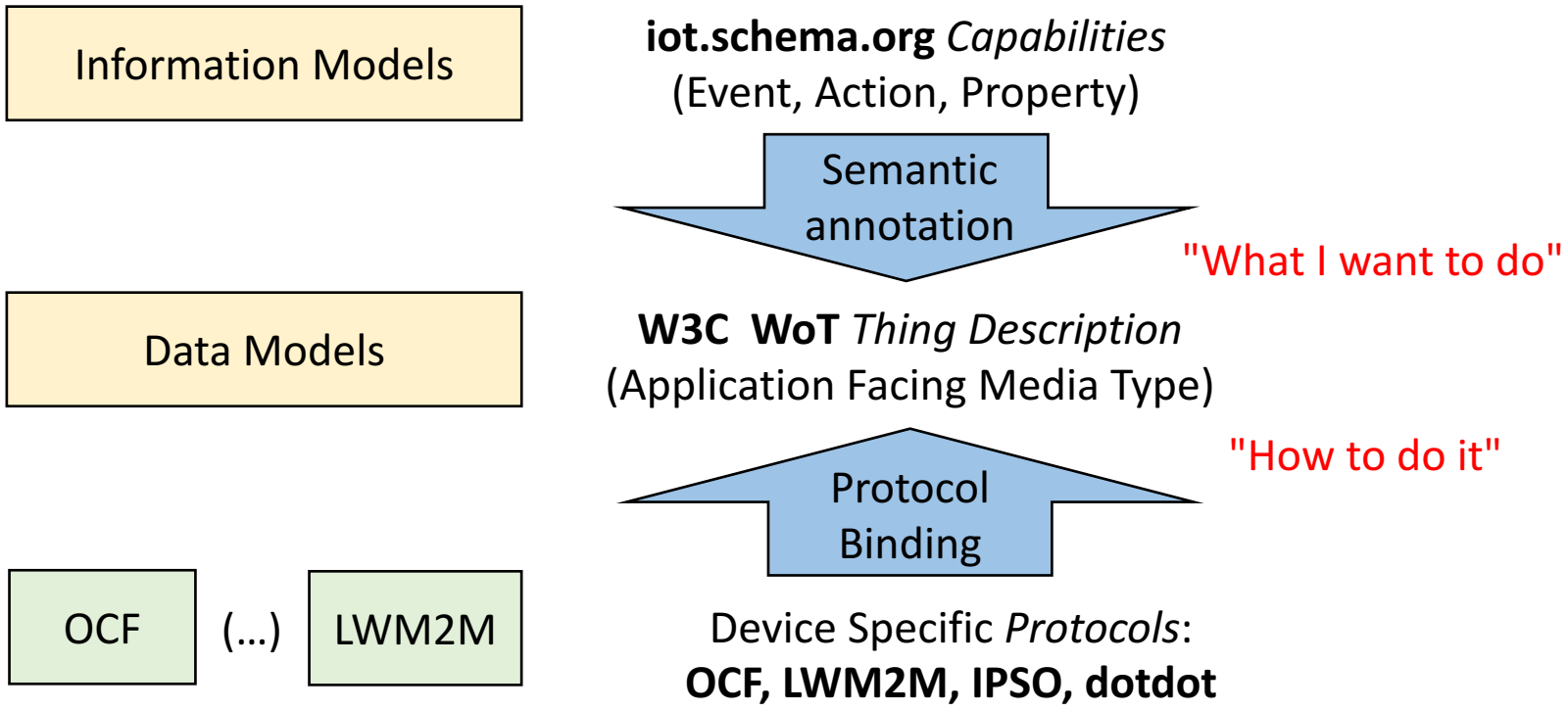
OCF Temperature Resource Representation using IPSO identifiers

```
GET /3303/0
{
 "5700": 27,
 "5701": "Cel",
 "5601": 22,
 "5602": 40,
 "5603": 0 ,
 "5604": 100
}
```

De-coupled approach

- "What to do" and "How to do it" can be decoupled
- "What to do" can be defined in a protocol-neutral and format-neutral way
  - Like ontologies and vocabularies
- "How to do it" can be defined in a way that takes advantage of network interoperability and accommodates diverse formats and interaction models
  - Like hypermedia controls

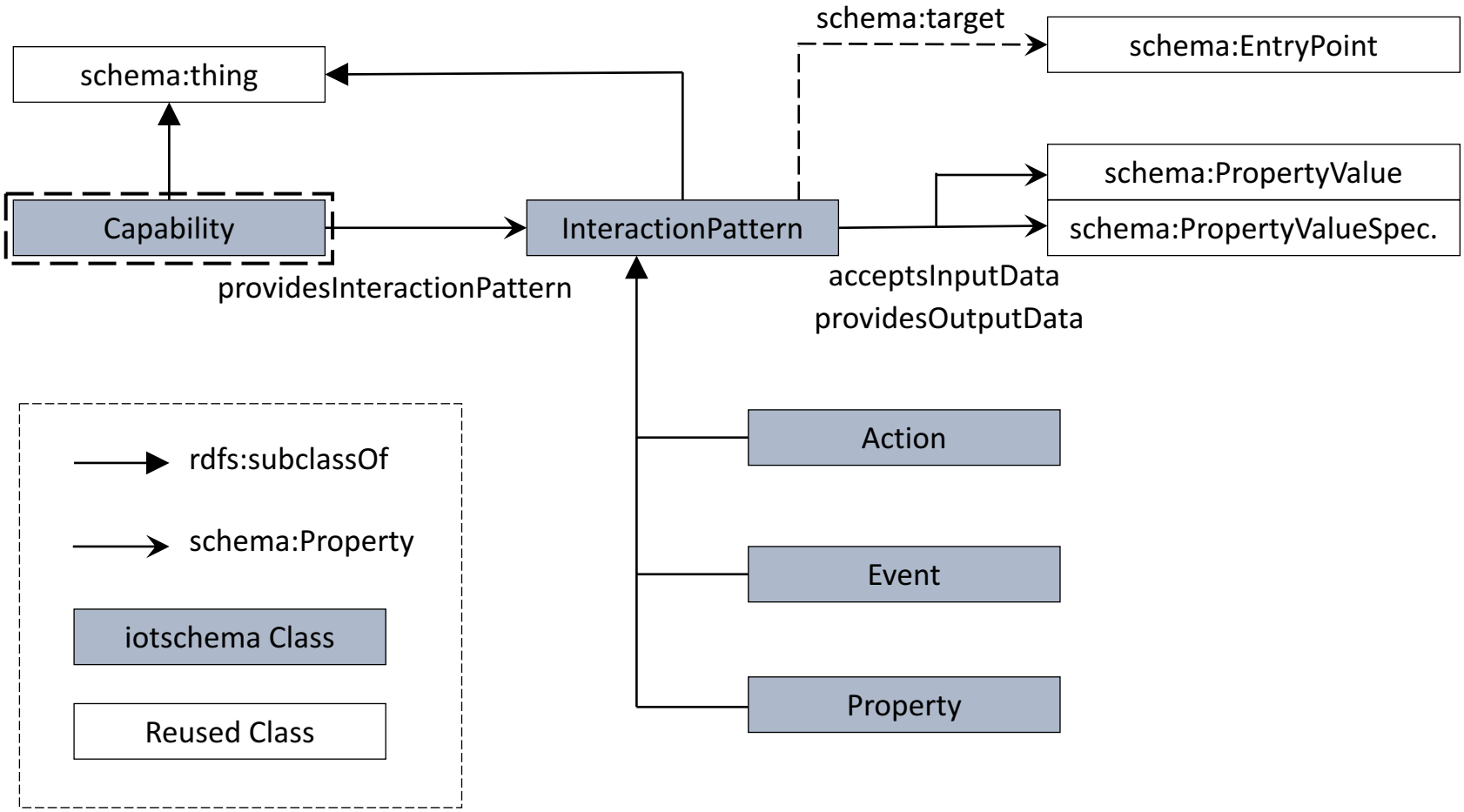
Layered Scope in Data Models and Information Models



iotschema

- Patterned after schema.org
- RDF definitions for connected things
- Start with common definitions for the common affordances of things
- Apply a simple meta model that captures existing models and functionality
- Create a venue where specific definitions may be developed for vertical market applications

iotschema Capability Pattern



Examples of Capabilities

Thing	Capabilities	Properties	Type	Actions
Motion Sensor	Motion Sensing	Motion	Boolean	(read)
Temperature	Temperature Sensing	Temperature	Number	(read)
Light	Switch	SwitchState	Boolean	TurnOn TurnOff
Light	Level Control	Level TransitionTime	Number Number	MoveToLevel MoveLevel StepLevel

Example iotschema Definition

```
"type": "TemperatureCapability",
"id": "iotschema:TemperatureCapability",
"subClassOf": "iot:Capability",
"description": "Level Sensing and Control Capability",
"providesInteractionPattern": {
    "type": "Property",
    "subClassOf": "iot:Property",
    "name" : "TemperatureProperty",
    "@id": "iot:TemperatureProperty",
    "providesOutputData": {
      "type": "schema:Number",
      "@id": "iot:TemperatureData",
      "schema:valueName": "temperatureData"
      "schema:unitCode": { "@id": "iot:TemperatureUnit" },
      "schema:minValue": "schema:Number",
      "schema:maxValue": "schema:NUmber"
    }
}
```

W3C WoT Thing Description

- Patterned after hypermedia controls
- Describes "What to do" in a common meta model format, without any application specific vocabulary
  - Semantic annotation from third party sources like iotschema
- Describes "How to do it" using a protocol binding technique
  - Hyperlinks, Payload templates, and transfer layer descriptions

TD Example – OCF Binding

```
{
  "@context": [
    "http://w3c.github.io/wot/w3c-wot-td-context.jsonld",
    "http://w3c.github.io/wot/w3c-wot-common-context.jsonld",
  {"iot": "http://iotschema.org/"}
  ],
  "base": "coap://example.net:5683/",
  "@type": [ "Thing", "iot:TemperatureCapability" ],
  "name": "Temperature Sensor",
  "interaction": [
    {
      "name": "Temperature",
      "@type": ["Property", "iot:TemperatureProperty"],
      "outputData":  {
        "type": "object",
        "field": [
          {
            "name": "temperature",
            "@type": ["iot:TemperatureData"],
            "type": "number",
            "minimum": -50,
            "maximum": 100,
            "unit": "Celsius"
          }
        ]
      }
```

TD Example – OCF Binding

```
"form": [
  {
    "href": "temperature?rt=oic.r.temperature&if=oic.if.s",
    "mediatype": "application/vnd.ocf+cbor",
    "rel": "td:getProperty",
    "coap:methodName": "coap:get",
    "coap:methodCode": "0.01",
    "coap:MessagelHeader": [
      {
        "coap:fieldName": "Accept",
        "coap:fieldCodeNumber": 17,
        "coap:fieldValue": 10000
      },
      {
        "coap:fieldName": "OCF-Accept-Content-Format-Version",
        "coap:fieldCodeNumber": 2049,
        "coap:fieldValue": "1.1.0"
      }
    ]
  },
```

TD Example – IPSO Binding

```
"name": "Temperature",
"@type": ["Property", "iot:TemperatureProperty"],
"outputData":  {
  "type": "array",
  "observable": "true",
  "item": [
    {
      "type": "object",
      "field": [
        {
          "name": "n",
          "value": "5700"
        },
        {
          "name": "v",
          "value": {
            "@type": ["iot:TemperatureData"],
            "type": "number",
            "minimum": -100,
            "maximum": 150,
            "unit": "Fahrenheit"
          }
        }
      ]
    }
  ]
```

TD Example – IPSO Binding

```
"form": [
  {
    "href": "3303/0",
    "mediatype": "application/json",
    "rel": "td:getProperty",
    "http:methodName": "http:get"
  },
  {
    "href": "3303/0",
    "mediatype": "application/json",
    "rel": "td:setProperty",
    "http:methodName": "http:post"
  },
  {
    "href": "mqtt://example.net:1883/3303/0",
    "rel": "td:observe",
    "mqtt:methodName": "mqtt:subscribe"
  },
  {
    "href": "mqtt://example.net:1883/3303/0",
    "rel": "td:observeCancel",
    "mqtt:methodName": "mqtt:unsubscribe"
  }
]
```